Инженерный вестник Дона, №6 (2022)
ivdon.ru/ru/magazine/archive/n6y2022/7758

Формирование требований к тренажеру по составлению прототипа функции на языке Си

Д.В. Литовкин, К.С. Кулюкин, О.А. Сычев, Е.И. Ромах

Волгоградский государственный технический университет, Волгоград

Аннотация: Одним из важных навыков в программировании является выделение подпрограммы (функции) как части кода и определение ее интерфейса (заголовка или прототипа). В данной работе описаны шаги, необходимые для составления прототипа функции, и типовые ошибки, совершаемые обучающимися. Для эффективного освоения данного навыка важно формировать индивидуальную обратную связь с обучающимся, чтобы он мог своевременно понимать допущенные им ошибки. Существующие автоматизированные подходы и инструменты не могут обеспечить достаточную информативную обратную связь при составлении прототипа функции, т.к. не учитывают предметную область решаемой задачи. Предлагается создать тренажер, который должен удовлетворять следующим требованиям: а) воспроизводить многошаговый процесс составления прототипа функции и оценивать результаты на каждом шаге, выявляя разнотипные ошибки и генерируя подсказки; б) допускать вариативность составления прототипа функции; в) обладать знаниями о предметной области, для которой составляется прототип функции, а также о правилах используемого языка программирования.

Ключевые слова: тренажер, требования к тренажеру, обучение программированию, прототип функции, задача со сложным результатом, многошаговая задача, дистанционное обучение, смешанное обучение, автоматизированная проверка ответа обучающегося.

Благодарности: *Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20–07–00764.*

Введение

Обучение программированию, прежде всего, связано с написанием программного кода и развитием соответствующих навыков. Одним из важных навыков в программировании является выделение подпрограммы (функции) как части кода и определение ее интерфейса (заголовка или прототипа) [1]. Для овладения этим навыком важно своевременно выявлять ошибки обучающегося и предоставлять ему обратную связь, чтобы помочь ему найти и исправить ошибки. Обеспечить полноценную обратную связь, реализуемую учителем, не всегда возможно, т.к. для этого требуются большие затраты труда и времени преподавательского состава. Помочь в решении этой проблемы могут интеллектуальные тренажеры.

Интеллектуальный тренажер - это компьютерная система, направленная на обеспечение немедленной и индивидуальной обратной связи ученику, обычно без вмешательства учителя [2]. Она воспроизводит индивидуальный учебный процесс ученика и учителя, улучшая понимание и восприятие [3, 4].

В этой статье мы рассматриваем задачу составления прототипа функции (на языке программирования Си) и требования к тренажеру, который бы обеспечивал индивидуальную обратную связь для обучающегося при решении этой задачи.

Задача составления прототипа функции

Обучающемуся необходимо составить прототип (заголовок) функции на языке программирования Си, для которой сформулировано ее предназначение. Предназначение описывает действие, выполняемое функцией над сущностями предметной области решаемой задачи и желаемый результат этого действия. Пример задания: «Напишите прототип функции, которая находит признак пересечения первого отрезка и второго отрезка на числовой оси. Примечание: границы отрезков заданы целыми числами в диапазоне [-10^10;10^10]».

Такая задача требует последовательного решения следующих подзадач:

1) Выделение действия, выполняемого функцией, из текстовой формулировки задания;

2) Выделение элементов данных (сущностей предметной области), участвующих в задаче, и определение их свойств:

2.1) Выделение элементов данных из текстовой формулировки задания;

2.2) Определение для каждого элемента данных его типа данных на концептуальном уровне;

2.3) Определение для каждого элемента данных его направления: входной, выходной или обновляемый элемент данных.

3) Определение имени функции на языке программирования;

4) Представление всех элементов данных в виде параметров или возвращаемых значений функции:

4.1) Выбор для каждого элемента данных способа его представления в памяти компьютера: скалярное значение, коллекция значений, сложная сущность с несколькими характеристиками или последовательность отдельных компонентов;

4.2) Определение для каждого элемента или компонента данных способа его передачи в функцию или возврата из функции: входной, выходной или обновляемый параметр, или возвращаемое значение;

4.3) Выбор типов данных языка программирования для каждого параметра и возвращаемого значения;

4.4) Определение имен параметров функции и их порядка.

5) Запись прототипа функции на языке программирования.

Пример составления прототипа для задачи, сформулированной выше, показан в таблице №1. Необходимо отметить, что при составлении прототипа функции возможно получить несколько приемлемых решений. Вариативность решений обеспечивается за счет:

1) наличия различных допустимых способов представления элементов данных в памяти компьютера. Например, отрезок числовой оси можно представить, как сущность с двумя характеристиками, как два отдельных компонента (показано в примере) и как коллекцию из двух значений;

2) наличия различных способов возврата из функции выходных и обновляемых элементов (компонентов) данных. Например, признак пересечения отрезков может быть реализован как возвращаемое значение или как выходной параметр функции;

3) возможности использовать взаимозаменяемые типы данных языка программирования для представления одного и того же параметра или возвращаемого значения;

4) различного именования функции и ее параметров.

Таблица №1

Пример пошагового составления прототипа функции на языке Си

№ под-задачи	Результат решения подзадачи
1	Действие: «Найти признак пересечения отрезков»
2	Входные данные: 1) «Первый отрезок» – сложная сущность, имеющая характеристики «левая граница» и «правая граница», которые представлены целыми числами в диапазоне [-10^10;10^10]; 2) «Второй отрезок» – аналогично «Первому отрезку». Выходные данные: 1) «Признак пересечения отрезков» – логическое значение.
3	has_intersection
4	Входные параметры: 1) long first_segment_left_border - левая граница первого отрезка; 2) long first_segment_right_border - правая граница первого отрезка; 3) long second_segment_left_border - левая граница второго отрезка; 4) long second_segment_right_border - правая граница второго отрезка. Возвращаемое значение: 1) int – признак пересечения отрезков.
5	int has_intersection(long first_segment_left_border, long first_segment_right_border, long second_segment_left_border, long second_segment_right_border) ;

Таким образом, задачу составления прототипа функции можно отнести к многошаговой задаче со сложным результатом [5]. При выполнении указанных подзадач обучающийся может совершать различные ошибки, которые приведут к неработающему, не эффективному или к плохо читаемому прототипу функции. На шагах 1) и 2) возникают ошибки анализа предметной области, а именно: а) непонимание, что должна делать функция,

или неверное определение действия, выполняемого функцией; б) неверное или неполное выделение элементов данных, участвующих в задаче; в) неверное или неточное определение для элемента данных его типа данных на концептуальном уровне; например, отрезок числовой оси представляется одним числовым значением, а не сложной сущностью с двумя числовыми характеристиками; в) неверное определение для элемента данных его направления. Ошибки анализа приводят к неработающему и плохо читаемому прототипу функции.

На шагах 3) и 4.4) возникают ошибки именования функции и ее параметров, а именно: а) имя функции не отражает предназначение функции; б) имя параметра не отражает, что параметр хранит в себе; в) двусмысленность имен; г) имя не соответствует стандарту кодирования или одновременно используются несколько стандартов кодирования. Ошибки именования приводят к плохо читаемому прототипу функции, что усложняет ее понимание и использование.

На шагах 4.1) – 4.3) возникают ошибки проектирования, связанные с представлением элементов данных в виде параметров и возвращаемых значений функции, а именно:

– представление элемента данных в памяти компьютера избыточно или недостаточно. Например, отрезок числовой оси задается сущностью из трех характеристик: левая граница, правая граница и длина отрезка;

– у языка программирования имеются ограничения по типу и количеству возвращаемых значений, и они нарушаются для выбранного представления элемента данных. Например, в большинстве языков программирования у функции может быть только одно возвращаемое значение;

– неверно определено направление параметра: входной, выходной или обновляемый параметр;

– слишком большое количество параметров, что ухудшает читаемость прототипа функции;

– не все элементы и компоненты данных представлены в виде параметров и возвращаемых значений; имеются избыточные параметры и возвращаемые значения, которые не соответствуют никаким элементам и компонентам данных;

– тип данных у параметра или возвращаемого значения ошибочен, т.к. противоречит представлению элемента данных в памяти компьютера.

Ошибки проектирования приводят к неработающему прототипу функции.

При условии, что все предыдущие подзадачи решены, на шаге 5) возможны только синтаксические ошибки, связанные с незнанием правил языка программирования или опечатками, например, отсутствует тип данных перед параметром функции.

Для составления прототипа функции требуется значительное количество разнообразных умений, поэтому начинающие программисты часто делают ошибки; им необходима обратная связь в процессе решения данной задачи.

Существующие подходы и инструменты, обеспечивающие автоматическую обратную связь при составлении прототипа функции

Наиболее доступным инструментом является статический анализатор кода (например, CppCheck, PVS-Studio), который анализирует программу без ее выполнения, путем просмотра всех потенциальных путей ее исполнения [6]. Анализатор оперирует законами языка программирования и не учитывает предметную область решаемой задачи, поэтому он может обнаружить в прототипе функции только синтаксические ошибки на шаге 5). Более того, при наличии значительных ошибок в прототипе функции

анализатор интерпретирует строку кода как другую синтаксическую конструкцию и выдает список ошибок, которые малоприменимы к прототипу функции.

Другой подход к обнаружению ошибок и генерации подсказок – это использование эталонного ответа, который учитывает предметную область решаемой задачи и правила языка программирования. Примером такого подхода является тестовый вопрос CorrectWriting [7] в системе дистанционного обучения Moodle. В нем реализованы задачи на составление прототипа функции, использующиеся в учебном процессе ВолгГТУ. Данный тип вопроса представляет собой вопрос в открытой форме, где ответ студента сопоставляется с эталонным ответом, заданным в виде последовательности лексем естественного языка или языка программирования. Для прототипа функции лексемами будут ключевые слова (типы данных и их модификаторы), имена функции и ее параметров, а также символы разделителей. Проверка ответа обучающегося производится путем сопоставления порядка лексем ответа с лексемами эталонного ответа. Если ответ обучающегося не соответствует эталонному, CorrectWriting выполняет поиск максимальной общей подпоследовательности, а затем генерирует список ошибок следующего типа: отсутствие лексемы, лишняя лексема, лексема находится не на своем месте, опечатка в лексеме. При этом в сообщении об ошибке используется смысловое описание лексемы. Также отличительной особенностью вопроса CorrectWriting является то, что он может не только определять правильность ответа, но и генерировать следующие типы подсказок: подсказать местоположение лексемы словесно и в виде изображения; показать лексему, соответствующую ее описанию.

Применительно к проектированию и написанию прототипа функции тестовый вопрос CorrectWriting имеет следующие недостатки. Он не допускает вариативность прототипа функции. Чтобы прототип функции был

однозначным в формулировке решаемой задачи необходимо явно указывать имя функции, ее параметры и возвращаемое значение. При такой постановке задачи обучающийся тренирует навыки, связанные с шагами 4.3) и 5), остальные навыки не тренируются.

Вопрос CorrectWriting оперирует только последовательностями лексем и не учитывает в явной форме предметную область решаемой задачи и законы языка программирования. Смысловая нагрузка каждой лексемы задается преподавателем путем задания ей смыслового описания в эталонном ответе. В результате для каждого экземпляра вопроса необходимо повторно задавать описание для однотипных лексем.

Ни один из рассмотренных подходов и инструментов не может обеспечить достаточную информативную обратную связь для тренировки навыков по составлению прототипа функции, т.к. не учитывает предметную область решаемой задачи.

Существуют тренажеры, которые одновременно учитывают предметную область решаемой задачи и правила языка программирования, например, Problets [8]. Чаще всего, в них используется декларативный анализ кода [9]. Однако такие тренажеры специализированы для решения конкретного типа задач, и нами не найдено ни одного, который мог бы использоваться для составления прототипа функции.

Требования к тренажеру

Для повышения доступности обратной связи для обучающихся при составлении прототипа функции с сохранением качества обратной связи предлагается использовать тренажер специального вида.

Тренажер должен обеспечивать пошаговое составление прототипа функции в виде последовательности подзадач, аналогичных приведенным выше. Пошаговая процедура должна обеспечивать постепенное приближение

к конечному результату (прототипу функции) с верификацией промежуточных результатов. При этом каждая подзадача не должна требовать от обучающегося множества навыков для ее решения (лучше если их будет один или два).

Тренажер должен оценивать результаты на каждом шаге, выявляя ошибки анализа предметной области решаемой задачи и ошибки проектирования, которые рассмотрены выше. Ошибки именования функции и ее параметров выявлять пока не планируется, т.к. существует большая вариативность имен.

При обнаружении ошибки тренажер должен генерировать подсказку, удовлетворяющую следующим требованиям: а) подсказка должна помочь обучающемуся осознать ошибку; б) подсказка должна использовать терминологию той предметной области, для которой создается прототип функции; в) подсказка должна использовать терминологию используемого языка программирования.

Тренажер должен допускать вариативность составления прототипа функции. Обучающийся может выбрать допустимый способ представления элемента данных в памяти компьютера, допустимый способ возврата из функции выходного или обновляемого элемента (компонента) данных, использовать взаимозаменяемые типы данных. Как следствие, решения, принятые обучающимся на предыдущих шагах, должны учитываться тренажером для генерации подзадач на последующих шагах.

Выводы

Задача составления прототипа является многошаговой процедурой и требует от обучающегося различных умений, что обуславливает возникновение разнотипных ошибок: ошибок анализа предметной области решаемой задачи, ошибок проектирования параметров и возвращаемого

значение функции, а также ошибок именования функции и ее параметров. Автоматическое обнаружение таких ошибок и генерация необходимых подсказок позволит повысить доступность обратной связи для обучающихся. В настоящее время не существует программных средств, которые бы выявляли все типы ошибок при составлении прототипа функции. Нами сформулированы требования к такому тренажеру. Чтобы тренажер умел определять разнотипные ошибки, он должен обладать знаниями о предметной области, для которой составляется прототип функции, а также обладать знаниями о правилах языка программирования в части составления прототипа функции. Для решения таких задач подходят тренажеры, основанные на ограничениях [10].

В дальнейшем планируется выполнение следующих задач: а) разработка сценария тренажера в виде последовательности подзадач; б) выбор методов оценки ответов обучающегося и генерации подсказок; в) проектирование структуры тренажера; г) создание тренажера; д) апробация тренажера в учебном процессе.

Литература

1. Deitel, P. and H. Deitel, 2013. C for Programmers with an Introduction to C11. Pearson. – 544 p.

2. Mutter, S. and J. Psotka, 1998. Intelligent Tutoring Systems: Lessons Learned. Mahwah: Hillsdale, NJ : L. Erlbaum Associates. 584 p.

3. Rathore, A.S., Arjaria, S.: Intelligent Tutoring System, pp. 121–144 (01 2020). doi.org/10.4018/978-1-7998-0010-1.ch006/.

4. Молчанов О. Е., Васильев А. С., Белая Т. И. Компьютерный тренажер-эмулятор учебной цифровой вычислительной машины // Инженерный вестник Дона, 2015, № 2 (часть 2) URL: ivdon.ru/ru/magazine/archive/n2p2y2015/3046/.

5. Латыпова В. А. Сложные открытые задачи в смешанном и дистанционном автоматизированном обучении // Инженерный вестник Дона, 2015, № 3 URL: ivdon.ru/ru/magazine/archive/n3y2015/3211/.

6. Аветисян А.И., Белеванцев А.А., Чукляев И.И. Технологии статического и динамического анализа уязвимостей программного обеспечения. Вопросы кибербезопасности. №3 (4) июль-сентябрь 2014 г., стр. 20-28.

7. Sychev, O.A. and D.P. Mamontov, 2018. Automatic error detection and hint generation in the teaching of formal languages syntax using correctwriting question type for moodle LMS. 3rd Russian-Pacific conference on computer technology and applications, RPC 2018 Vladivostok, Institute of Electrical and Electronics Engineers Inc. Date Views 16.06.2022. URL: ieeexplore.ieee.org/document/8482125.

8. Expression Tasks for Novice Programmers, Matthias Laengrich, Joerg Schulze, Amruth N. Kumar, Proceedings of Frontiers in Education Conference (FIE 2015), El Paso, TX, 10/21-24/2015. Pp. 300-307.

9. Towards Ontology-Based Program Analysis / Y. Zhao, Ch. Liao, X. Shen. – Livermore, 1998. – 25 p.

10. Mitrovic, A., K.R. Koedinger and B. Martin, 2003. A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling. International Conference on User Modeling, Lecture Notes in Computer Science Date Views 16.06.2022. URL: link.springer.com/chapter/10.1007/3-540-44963-9_42.

References

1. Deitel, P. and H. Deitel, 2013. C for Programmers with an Introduction to C11. Pearson. 544 p.

2. Mutter, S. and J. Psotka, 1998. Intelligent Tutoring Systems: Lessons Learned. Mahwah: Hillsdale, NJ : L. Erlbaum Associates. 584 p.

3. Rathore, A.S., Arjaria, S.: Intelligent Tutoring System, pp. 121–144 (01 2020). doi.org/10.4018/978-1-7998-0010-1.ch006/.

4. Molchanov O.E., Vasil'ev A.S., Belaya T.I. Inzhenernyj vestnik Dona, 2015, № 2. URL: ivdon.ru/ru/magazine/archive/n2p2y2015/3046.

5. Latypova V.A. Inzhenernyj vestnik Dona, 2015, № 3. URL: ivdon.ru/ru/magazine/archive/n3y2015/3211/.

6. Avetisyan A.I., Belevanzev A.A., Chuklyaev I.I. Tehnologii staticheskogo i dinamicheskogo analiza ujazvimostej programmnogo obespechenija. Voprosy kiberbezopasnosti. №3 (4). July - September, 2014. Pp. 20-28.

7. Sychev, O.A. and D.P. Mamontov, 2018. Automatic error detection and hint generation in the teaching of formal languages syntax using correctwriting question type for moodle LMS. 3rd Russian-Pacific conference on computer technology and applications, RPC 2018 Vladivostok, Institute of Electrical and Electronics Engineers Inc. Date Views 16.06.2022. URL: ieeexplore.ieee.org/document/8482125.

8. Expression Tasks for Novice Programmers, Matthias Laengrich, Joerg Schulze, Amruth N. Kumar, Proceedings of Frontiers in Education Conference (FIE 2015), El Paso, TX, 10/21-24/2015. Pp. 300-307.

9. Towards Ontology-Based Program Analysis / Y. Zhao, Ch. Liao, X. Shen. – Livermore, 1998. – 25 p.

10. Mitrovic, A., K.R. Koedinger and B. Martin, 2003. A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modeling. International Conference on User Modeling, Lecture Notes in Computer Science Date Views 16.06.2022. URL: link.springer.com/chapter/10.1007/3-540-44963-9_42.